

Docket No.: 61282-058

**PATENT**

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of	:	Customer Number: 20277
	:	
Masayoshi KODAMA, et al.	:	Confirmation Number:
	:	
Serial No.:	:	Group Art Unit:
	:	
Filed: January 30, 2004	:	Examiner:
	:	
For: MEMORY MANAGING SYSTEM AND TASK CONTROLLER IN MULTITASK SYSTEM		

**CLAIM OF PRIORITY AND  
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Mail Stop Patent Application  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

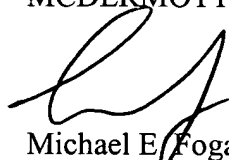
In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

**Japanese Patent Application No. 2003-282887, filed July 30, 2003**

A Certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY



Michael E. Fogarty  
Registration No. 36,139

600 13<sup>th</sup> Street, N.W.  
Washington, DC 20005-3096  
(202) 756-8000 MEF:prg  
Facsimile: (202) 756-8087  
**Date: January 30, 2004**



日 本 国 特 許 庁  
JAPAN PATENT OFFICE

61282-058  
Kodama et al.  
January 30, 2004  
McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日                      2 0 0 3 年    7 月 3 0 日  
Date of Application:

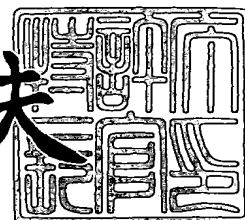
出 願 番 号                      特 願 2 0 0 3 - 2 8 2 8 8 7  
Application Number:  
[ST. 10/C]:                      [ J P 2 0 0 3 - 2 8 2 8 8 7 ]

出      願      人                      松 下 電 器 産 業 株 式 会 社  
Applicant(s):

2 0 0 3 年 1 0 月 1 7 日

特許庁長官  
Commissioner,  
Japan Patent Office

今 井 康 夫



出証番号    出証特 2 0 0 3 - 3 0 8 5 7 3 2

【書類名】 特許願  
【整理番号】 5037750003  
【提出日】 平成15年 7月30日  
【あて先】 特許庁長官殿  
【国際特許分類】 G06F 13/10  
【発明者】  
    【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内  
    【氏名】 小玉 将義  
【発明者】  
    【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内  
    【氏名】 小林 圭太  
【特許出願人】  
    【識別番号】 000005821  
    【氏名又は名称】 松下電器産業株式会社  
【代理人】  
    【識別番号】 100105647  
    【弁理士】  
    【氏名又は名称】 小栗 昌平  
    【電話番号】 03-5561-3990  
【選任した代理人】  
    【識別番号】 100105474  
    【弁理士】  
    【氏名又は名称】 本多 弘徳  
    【電話番号】 03-5561-3990  
【選任した代理人】  
    【識別番号】 100108589  
    【弁理士】  
    【氏名又は名称】 市川 利光  
    【電話番号】 03-5561-3990  
【選任した代理人】  
    【識別番号】 100115107  
    【弁理士】  
    【氏名又は名称】 高松 猛  
    【電話番号】 03-5561-3990  
【選任した代理人】  
    【識別番号】 100090343  
    【弁理士】  
    【氏名又は名称】 栗宇 百合子  
    【電話番号】 03-5561-3990  
【手数料の表示】  
    【予納台帳番号】 092740  
    【納付金額】 21,000円  
【提出物件の目録】  
    【物件名】 特許請求の範囲 1  
    【物件名】 明細書 1  
    【物件名】 図面 1  
    【物件名】 要約書 1  
    【包括委任状番号】 0002926

**【書類名】 特許請求の範囲****【請求項 1】**

割込み発生に応じて、CPU内情報を被割込みタスクのタスクスタックに格納して第1領域となし、該格納後のスタックポインタの値を割込み処理スタックの所定の第1位置に格納し、スタックポインタを前記割込み処理スタックの所定の第2位置にセットし、割込み処理を開始するマルチタスクシステムのスタック管理方法であって、

前記割込みスタック処理スタックの先頭アドレスを、特定タスクのタスクスタックにおける前記第1領域内の所定位置に一致させることを特徴とするスタック管理方法。

**【請求項 2】**

前記第1領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第2領域と、該第2領域の直後に配置され前記タスク制御情報以外のCPU内情報を格納する第3領域とからなり、

前記特定タスクのタスクスタックにおける第1領域内の所定位置は、該第1領域における前記第3領域の先頭アドレスであることを特徴とする請求項1記載のスタック管理方法。

**【請求項 3】**

前記割込み処理スタックの所定の第1位置は、該割込み処理スタックの先頭アドレスであることを特徴とする請求項1記載のスタック管理方法。

**【請求項 4】**

前記割込み処理スタックの所定の第2位置は、前記割込み処理スタックの所定の第1位置の直後に位置することを特徴とする請求項1記載のスタック管理方法。

**【請求項 5】**

前記第1領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第2領域と、該第2領域の直後に配置され前記タスク制御情報以外のCPU内情報を格納する第3領域とからなり、

前記特定タスクのタスクスタックにおける第1領域内の所定位置は、該第1領域における前記第3領域の先頭アドレスに所定のアドレス差分値を補正したアドレスであることを特徴とする請求項1記載のスタック管理方法。

**【請求項 6】**

前記アドレス差分値は、前記特定タスクにおいて使用されるCPU内情報を格納するために必要な領域を指定するように与えられるものであることを特徴とする請求項5記載のスタック管理方法。

**【請求項 7】**

前記アドレス差分値は、あらかじめ定数として与えられることを特徴とする請求項5記載のスタック管理方法。

**【請求項 8】**

前記アドレス差分値は、前記特定タスクの実行により設定されるものであることを特徴とする請求項5記載のスタック管理方法。

**【請求項 9】**

前記タスク制御情報は少なくとも、プログラムカウンタと、CPUの状態を表すプログラムステータス語（PSW）とを含むことを特徴とする請求項2記載のスタック管理方法。

**【請求項 10】**

前記特定タスクは、CPU内情報を使用せずに自己ループするアイドルタスクであることを特徴とする請求項1記載のスタック管理方法。

**【請求項 11】**

前記特定タスクは、低消費電力モードへの移行と復帰を制御するタスクであることを特徴とする請求項1記載のスタック管理方法。

**【請求項 12】**

前記第2領域を設けず、前記タスク制御情報および所定範囲の前記CPU内情報は被割込みタスクのタスクスタックに格納せずにタスクスタックと異なるメモリ領域に格納し、前

記所定範囲のCPU内情報以外のCPU内情報は前記第3領域に格納されることを特徴とする請求項2記載のスタック管理方法。

【請求項13】

割込み発生に応じて、CPU内情報を被割込みタスクのタスクスタックに格納して第1領域となし、該第1領域格納後のスタックポインタの値を割込み処理スタックの所定の第1位置に格納し、スタックポインタを前記割込み処理スタックの所定の第2位置にセットし、割込み処理を開始するマルチタスクシステムのスタック制御装置であって、

前記割込み処理スタックの先頭アドレスを、特定タスクのタスクスタックにおける前記第1領域内の所定位置に一致させる制御機構を備えることを特徴とするスタック制御装置。

【請求項14】

前記第1領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第2領域と、該第2領域の直後に配置され前記タスク制御情報以外のCPU内情報を格納する第3領域とからなり、

前記特定タスクのタスクスタックにおける第1領域内の所定位置は、該第1領域における前記第3領域の先頭アドレスとすることを特徴とする請求項13記載のスタック制御装置。

【請求項15】

前記第1領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第2領域と、該第2領域の直後に配置され前記タスク制御情報以外のCPU内情報を格納する第3領域とからなり、

前記特定タスクのタスクスタックにおける第1領域内の所定位置は、該第1領域における前記第3領域の先頭アドレスに所定のアドレス差分値を補正したアドレスとすることを特徴とする請求項13記載のスタック制御装置。

【請求項16】

前記アドレス差分値は、前記特定タスクにおいて使用されるCPU情報を格納するために必要な領域を示すアドレス差分値格納手段に保持されることを特徴とする請求項13記載のスタック制御装置。

【請求項17】

前記アドレス差分値は、あらかじめ定数として与えられることを特徴とする請求項13記載のスタック制御装置。

【請求項18】

前記アドレス差分値は、前記特定タスクの実行により設定されるものであることを特徴とする請求項13記載のスタック制御装置。

【請求項19】

請求項5から8のうちいずれか1項記載のスタック管理方法、または請求項15から18のうちいずれか1項記載のスタック制御装置における前記アドレス差分値を自動的に生成するコンパイラ。

**【書類名】 明細書****【発明の名称】** マルチタスクシステムにおけるメモリ管理方式およびタスク制御装置**【技術分野】****【0001】**

本発明は、ソフトウェアのスタックメモリの管理方法、特にマルチタスクシステムを実行する場合のスタックメモリ使用量を削減するプログラム構造に関する。

**【背景技術】****【0002】**

プログラムにおける制御の複雑化の進展により、コンピュータによる仕事の単位であるタスクを一度に2つ以上処理することができるマルチタスクシステムが一般的になってきている。このマルチタスクシステムを用いることによって、複数のタスクを効率よく切替えて実行することができる。

**【0003】**

図3は、従来のマルチタスクシステムにおける、RAMの使用状況の一例を模式的に示した図である。図3における番号がふられた領域は、各タスクで使用するスタックメモリ領域（以下、スタックという）を示している。すなわち、RAM上にはマルチタスクシステムによって実行される複数のタスク1、タスク2、・・・、タスクnの複数のスタックと、アイドルタスク用スタック、割込み処理専用のスタックとが構築されることとなる。

**【0004】**

各スタックは各々、スタック301、311、321、戻りPC保存領域302、312、322、PSW保存領域303、313、323及びCPUレジスタ保存領域314、324、304によって構成されている。また、割込み処理用スタックは、割込み発生時のタスクSPを保存しておく戻りSP保存領域305と、割込み処理シーケンスで使用する割込み処理用スタック領域306によって構成される。

**【0005】**

ここで、割込み処理とはタスク（通常処理）が実行されているときに、このタスクを一時的に中断して、例えばタイマ等の制御により一定の時間毎に実行される処理や、外部要因によって実行される処理などである。

**【0006】**

例えば、タスク1が動作している時に割込みが発生した場合には、スタック311を用いて実行されているタスク1は一時的に中断される。このとき、現在のPCレジスタの値は戻りPC保存領域312に保存され、現在のPSWレジスタの値はPSW保存領域313に保存される。さらに、タスク1で使用されていたCPUレジスタの値をCPUレジスタ保存領域314に保存する。次に、タスク1のSPレジスタの値を割込み専用スタック領域内の戻りSP保存領域305に保存し、SPレジスタの値をSP保存領域305と割込みで使用するスタック領域306の境界、すなわちスタック領域306のボトム領域を指すように設定することで、スタック領域をタスクスタックから割込み処理用スタックへ切替える。

**【0007】**

割込み処理が終了するときは、SP保存領域305に保存しておいた値をSPレジスタに設定することでタスクスタックに切替える。その後は、CPUレジスタ保存領域314に保存しておいた値を各CPUレジスタへ設定し、PSW保存領域313と戻りPC保存領域312に保存しておいた値も同様にPSWレジスタとPCレジスタに復帰することで、元のタスク1へ復帰することができる。このような構成にすることにより、マルチタスクシステムによって複数のタスクを実行中に所定の割込み処理を行なうことができるようになっている（例えば、特許文献1参照）。

**【特許文献1】** 特開平08-123698号公報**【発明の開示】****【発明が解決しようとする課題】**

**【0008】**

しかしながら、図3に示したスタック構造では、割込み発生時に保存するCPUレジスタの退避容量が、タスクの処理内容に関わらず一意に決まったものであり、この中には保存する必要のないレジスタも多く含まれている可能性があり、不必要にメモリを消費している。これに対しては、例えば割込み処理の先頭部分で、保存すべきCPUレジスタの種類を判別するような方法があるが、割込み応答性能を悪化させてしまうという問題点を有している。

**【課題を解決するための手段】****【0009】**

本発明は、このような問題点を解決するもので、マルチタスクシステムを実行するn個のタスクスタックと、前記n個の各タスクスタックに対して共有されて機能する割込み処理用スタックと、割込み処理が発生したときに実行中のタスクスタックに退避されるPC、PSW、CPUレジスタとがあり、前記割込み処理用スタック領域は前記n個のタスクスタックのいずれか1つのタスクスタックと共有して使用される。上記割込み処理が発生したときに現在のタスクスタックにPC、PSW、CPUレジスタの値を退避したのちにスタックポインタを割込み処理側に切替え、上記割込み処理が終了したときにはスタックポインタをタスクスタックに切替えた後に、上記タスクスタック上に保存されたPC、PSW、CPUレジスタの値をタスクスタックから復帰させ、タスクの動作を再開させるようにする。

**【発明の効果】****【0010】**

以上の説明により、この発明の効果として、リアルタイムOSを用いたシステムのタスクスタック領域と割込みスタック領域とを重複して使用することで、システム全体でのメモリ使用量を削減できるということが言える。したがって、小容量のメモリでマルチタスクシステムを稼働させることができるという利点がある。

**【発明を実施するための最良の形態】****【0011】**

以下に、本発明の実施の形態について、図面を参照しながら説明する。

**(実施の形態1)**

図1は、本発明のマルチタスクシステムにおけるスタックの使われ方を示す図である。このマルチタスクシステムは、図2に示す従来のマルチタスクシステムに比べて割込み処理用スタック107の配置場所に特徴がある。

**【0012】**

図2に示す従来のマルチタスクシステムは、3つのタスク102、104、106と1つの割込み処理プログラム108がオペレーティングシステム（以下OSという）109上で動作しており、それぞれのタスクやプログラムには独立にスタック領域101、103、205、207が用意されている。また、3つのタスクのうち、最も優先度の低いタスクをアイドルタスク106と呼び、このアイドルタスクが動作しているときは、システムが処理すべきタスク処理がない状態であり、外部からの割込み要求が入ってくるのを待っている状態を示している。

**【0013】**

一方、図1に示す本発明のマルチタスクシステムも同様に、3つのタスク102、104、106と1つの割込み処理プログラム108がOS109の上で動作しており、それぞれのタスクやプログラムには独立にスタック領域101、103、105、107が用意されている。また、3つのタスクのうち、最も優先度の低いアイドルタスク106も同様に備えている。ただし、割込み処理用のスタック107は、アイドルタスク用スタック領域105に対して重なるように配置されており、割込み処理時のスタック領域はアイドルタスク処理で使用されたスタック領域を上書き（スタック破壊）しながら動作する。

**【0014】**

図3は、従来のマルチタスクシステムにおいて、通常のタスクスタック領域が2つ、ア

アイドルタスク用のスタック領域が1つ、割込み処理用スタック領域が1つある場合のスタックの内容を示す図である。

#### 【0015】

図3において、まずタスク1が動作しているときは、スタックポインタ（以下SP）がタスク1用のスタック領域311を指しており、タスク1はこの領域を使用しながらプログラムを処理している。この状態で割込みが発生し、CPUがその割込みを受理すると、ハードウェア的にプログラムカウンタ（以下、PC）とプロセッサステータスワード（以下、PSW）の内容をそれぞれスタック領域312、313に保存する。SPの値は、自動的にPCとPSWのサイズ分だけ減算される。次に、CPUはPCの値を割込み処理プログラムの先頭位置へ移動させ、OS割込み入口処理が実行される。

#### 【0016】

図5は、従来のマルチタスクシステムにおける、タスク動作中に割込みを受理したときのOS割込み入口処理のフローチャートである。以下では、タスク1の動作中に割込みが発生した場合について説明する。まず、ステップS501でCPUレジスタの値をスタック領域314へ保存する。CPUレジスタの数は、CPUの種類によって異なるが、最も簡単な実装はCPUが搭載している全てのCPUレジスタを保存することである。次に、ステップS502で、現在のSPの値（これは該当するタスクスタックの最終アドレスと等しい）を割込み処理用スタック領域のボトム位置にある戻りSP保存領域305に保存する。これは、割込み処理終了時にSPの指す場所を再びタスクスタックへ復帰させるために必要な処理である。次に、ステップS503では、割込み処理用スタック領域306のボトムアドレスをSPに代入して、スタックをタスクスタック314から割込み処理専用スタック306へ切替える。このとき、タスクへの戻りSP保存領域305を上書きしないように、SPの値を設定することに注意する。最後に、ステップS504にて、アプリケーションによって定義された割込みハンドラ関数をコールし、ステップS505にてアプリケーションの割込みハンドラプログラムを実行する。ステップS505からは、関数リターンという形でOS割込み入口処理へ復帰し、その後はOS割込み出口処理へと移行する。

#### 【0017】

図8は、割込み処理を終了しタスクへ復帰するときのOS割込み出口処理のフローチャートである。OS割込み出口処理では、まず、ステップS801でタスクへの戻りSP保存領域305に保存されている値をSPへ復帰させる。これにより、割込み処理用スタック領域306は未使用状態になる。一般的なOSでは、ステップS802にて遅延ディスパッチ処理を行なう。これは、割込み処理からタスクへ復帰する際に、タスクスイッチを行なうべきかどうかを判定し、タスクスイッチを行なう必要があれば、SPの値を現在のタスク1から他タスク（例えば、タスク2）の適当なスタック番地を指すように変更することで行なう。ただし、本発明では遅延ディスパッチの処理内容に関しては、本発明の主題とするところではないので、特に言及しないこととする。ステップS803では、ステップS501で保存しておいたCPUレジスタの値をスタック領域314から各CPUレジスタに復帰し、最後にステップS804にて割込みによって中断されていたタスク処理に復帰する。このとき、CPUが用意している割込み処理から復帰する命令を実行することで、PSWとPCの内容をスタックから復帰する。

#### 【0018】

タスク動作中に割込みを受理し、OS割込み入口処理から、出口処理を経由してタスクへ復帰するステップは、上記で説明したタスク1でなくても、図3で示したタスク2または、アイドルタスクでも全く同様である。つまり、図3においてタスク1が使用している311～314のスタック領域を、そのまま321～324、または301～304のスタック領域として読み替えればよい。

#### 【0019】

図4は、本発明のマルチタスクシステムにおいて、通常のタスクスタック領域が2つ、アイドルタスク用のスタック領域が1つある場合のスタックの中身を示す図である。



**【0020】**

図4において、まずタスク1が動作しているときはSPが311に示すタスク1用のスタック領域を指しており、タスク1はこの領域を使用しながらプログラムを処理している。この状態で割り込みが発生し、CPUがその割り込みを受理すると、ハードウェア的にPCとPSWの内容をそれぞれスタック領域312、313に保存する。このとき、SPの値は、自動的にPCとPSWのサイズ分だけ減算される。次に、CPUはPCの値を割り込み処理プログラムの先頭位置へ移動させ、OS割り込み入口処理が実行される。

**【0021】**

図6は、タスク動作中に割り込みを受理したときの割り込み入口処理のフローチャートを示している。OS割り込み入口処理では、まずステップS501で、CPUレジスタの値をスタック領域314へ保存する。CPUレジスタの数はCPUの種類によって異なるが、最も簡単な実装はCPUが搭載している全てのCPUレジスタを保存することである。次に、ステップS602で、現在のSPの値（これは該当するタスクスタックの最終アドレスと等しい）を割り込み処理用スタック領域のボトム位置にある戻りSP保存領域405に保存する。これは、割り込み処理終了時にSPの指す場所を再びタスクスタックへ復帰させるために必要な処理である。ここで、注意を要するのは、図5における従来のマルチタスクシステムと異なり、戻りSP保存領域405が、アイドルタスクのCPUレジスタ保存領域404に重なって配置されていることである。次に、ステップS603では、割り込み処理用スタック領域406のボトムアドレスをSPに代入して、スタックをタスクスタックから割り込み処理専用スタックへ切替える。このとき、タスクへの戻りSP保存領域405を上書きしないように注意する。最後に、ステップS504にて、アプリケーションによって定義された割り込みハンドラ関数をコールし、ステップS505にてアプリケーションの割り込みハンドラプログラムを実行する。ステップS505からは関数リターンという形でOS割り込み出口処理へと移行する。OS割り込み出口処理については、図8により説明した従来技術と同じ処理となるので、詳細の説明は省略する。

**【0022】**

ここで、タスク動作中に割り込みを受理し、OS割り込み入口処理からOS割り込み出口処理を経由してタスクへ復帰するステップは、上記で説明したタスク1でなくても、図4で示したタスク2についても全く同様である。つまり、図4においてタスク1が使用しているスタック領域311～314を、そのままスタック領域321～324として読み替えればよい。

**【0023】**

ただし、アイドルタスクが使用しているスタック領域に関しては事情が異なる。従来のマルチタスクシステムでは、アイドルタスクが使用しているスタック領域301～304は、前記タスク1のスタック領域311～314にそのまま置き換えることができるが、本発明の場合は、図4に示すスタック領域404と割り込み処理用スタック領域405および406とが重なって配置されているので、注意を要する。以下では、アイドルタスクが動作している時に割り込みを受理した場合の動作を詳細に説明する。

**【0024】**

まず、アイドルタスクが動作しているときはSPがアイドルタスクスタック領域401内の番地を指しており、アイドルタスクはこの領域を使用しながらプログラムを処理している（ただし、この領域は必ず存在するとは限らず、ゼロバイトの場合もある）。この状態で割り込みが発生し、CPUがその割り込みを受理すると、ハードウェア的にPCとPSWの内容をそれぞれ戻りPC保存領域402、PSW保存領域403に保存する。ここで、SPの値は自動的にPCとPSWのサイズ分だけ減算される。すなわち、PCとPSWがそれぞれ4バイトだった場合、スタックは8バイト分がそれらの保存に消費され、SPを8バイト分だけ減算したことに等しい。次に、CPUはPCの値を割り込み処理プログラムの先頭番地へ移動させ、ここでOS割り込み入口処理が実行される。アイドルタスクが動作しているときに割り込みを受理した場合のOS割り込み入口処理は図6に示すフローチャートに従って動作する。

**【0025】**

一方、OS割込み出口処理は、図8により説明した従来技術と同じ処理となるが、ステップS803で復帰するCPUレジスタの値は、既に上書きされているため、OS割込み入口処理時に保存しておいた値ではないことに注意を要する。

**【0026】**

図9は、実施の形態1におけるアイドルタスクの動作を表すフローチャートである。実施の形態1におけるアイドルタスクは、図9に示すように、自分自身への無限ループS901で構成されるよう実装されている。従って、図8におけるステップS803において、割込み処理内でPCとPSWを除く全てのCPUレジスタが不定の値に上書きされても、アイドルタスクのプログラムがCPUレジスタを全く使用せずに動作しているので、割込み復帰後も何ら問題は生じない。

**【0027】**

図12は、上記で説明した本発明のマルチタスクシステムを実現するタスク制御装置を示している。本タスク制御装置は、PC1205が現在実行中のプログラムの命令アドレス1209を指しており、この値はバスコントロールユニット（以下BCU）1212を通して、外部メモリ（ROM）1213から命令データ1210を取得する。命令データ1210は、命令実行制御部1206でデコード処理され、デコードされた命令の種別に応じて、タスク制御装置1200全体の動作を決定する。命令データが外部メモリへのデータ書き込みを行なう命令であった場合、命令実行制御部1206は、CPUレジスタ1201から必要なアドレス値を読み出して演算ユニット（以下ALU）1204に渡し、オペランドアドレス1207を、BCU1212を経由して外部メモリ（RAM）1214へアドレスデータとして渡す。同時に、CPUレジスタ1201の中からオペランドデータ1208を読み出し、BCU1212を経由して外部メモリ（RAM）1214へデータを書込む。なお、本タスク制御装置は、割込み制御部1211を搭載しており、周辺機能1215からの割込み要求を受理して、命令実行制御部1206へ通知し、現在実行中のプログラムを中断する機能を備えている。

**【0028】**

さらに本タスク制御装置は、割込み処理時のスタックアドレス制御のためにSP差分定数1250を備えている。あるタスクの実行中に割り込みが発生した場合は、SP1202の値をオペランドアドレス1207およびBCU1212を経由して外部メモリ（RAM）1214へタスクスタックのアドレスとして渡し、PC1205およびPSW1203の値をタスクスタックに保存した後、その分だけSPの値を更新する。続けて、図6のフローチャートに示すように、CPUレジスタの内容をタスクスタックに保存し、その分だけSPの値を更新する。

**【0029】**

次に、この最終のSP1202の値を割込み処理スタックのSP保存領域405に保存する。OS割込み入口処理プログラムあるいは割込み制御部1211には割込み処理スタックの重畳先としてアイドルタスクスタックのCPUレジスタ保存領域404の先頭アドレスが予め与えられている。また、SP差分定数1250には、アイドルタスクスタックのCPUレジスタ保存領域404の先頭アドレスに対する割込み処理スタックの先頭アドレスのアドレス差分値が与えられている。

**【0030】**

割込み処理スタックの重畳先、すなわちアイドルタスクスタックのCPUレジスタ保存領域の先頭アドレスがOS割込み入口処理プログラムあるいは割込み制御部1211から与えられ、これとSP差分定数1250の値をALU1204で演算し、これをオペランドアドレス1207およびBCU1212を経由して外部メモリ（RAM）1214へアドレスとして渡すことで、前記最終のSPの値を保存する。ここで、本実施の形態ではSP差分定数1250に0を与えることで、割込み処理スタックのSP保存領域405の先頭アドレスをアイドルタスクスタックのCPUレジスタ保存領域の先頭アドレスに一致させる。この先頭アドレスをSP保存領域分だけ更新してSPにセットすることで、割込み

処理スタックが使用可能になり割込み処理が先に進められる。

#### 【0031】

以上のように、割込み処理スタックをアイドルタスクのタスクスタックに重畳することにより、アイドルタスクのタスクスタックの使用されないCPUレジスタ保存領域を有効活用することができる。

#### 【0032】

(実施の形態2)

発明の実施の形態1で説明したマルチタスクシステムは、図9に示すような単純な無限ループのみで構成されたアイドルタスク以外にも、図10のフローチャートに示すような他の処理を含むようなアイドルタスクでも実現することができる。図10において、アイドルタスクプログラムはステップS1001に示すように、CPUの動作モードを低消費電力モードへ移行する処理を行なう。ここで、CPUが低消費電力モードへ移行した後は、割込みが受理されるまでその後の命令は実行されない。CPUに割込み要求が入り、低消費電力モードが解除されると、CPUは通常の動作モードに復帰すると同時に、図7のフローチャートに示したOS割込み入口処理を実行することとなる。

#### 【0033】

ここで、CPUの動作モードを低消費電力モードへ移行させる際に、CPUレジスタのうちの1つであるr0レジスタを使用するとする。この場合、割込み処理の前後でもr0レジスタの値は保存されていなければならない。

#### 【0034】

図11は、この場合のアイドルスタックと割込み処理用スタックの使用方法を表す図である。まず、アイドルタスクが動作しているときは、SPがアイドルタスク用のスタック領域1101を指しており、アイドルタスクはこの領域を使用しながらプログラムを処理している。上記プログラムが、r0レジスタを使用している最中に割込みが発生し、CPUがその割込みを受理すると、ハードウェア的にPCとPSWの内容をそれぞれスタック領域1102、1103に保存する。SPの値は、自動的にPCとPSWのサイズ分だけ減算される。CPUは、PCの値を割込み処理プログラムの先頭番地へ移動させ、図7に示すOS割込み入口処理が実行される。

#### 【0035】

図7における、OS割込み入口処理では、まずステップS701でCPUレジスタの値をスタック領域1107および1104へ保存する。この場合、ステップS801では、必ずr0レジスタをPSW保存領域1103のすぐ上にあるr0レジスタ保存領域1107に保存するようにする。その他のCPUレジスタに関しては保存する順番は問わない。次に、ステップS702で、現在のSPの値を割込み処理用スタック領域のボトム位置1105に保存する。実施の形態1と異なり、戻りSP保存領域1105は、r0レジスタ保存領域1107を上書きしないような位置になっていなければならないが、それ以外のCPUレジスタを保存するCPUレジスタ保存領域1104に対しては重なるように配置されている。ステップS703では、割込み処理用スタック領域1106のボトムアドレスをSPに代入して、スタックをアイドルタスクスタックから割込み処理用スタック領域1106へ切替える。割込み処理用スタック領域1106は、先ほどのCPUレジスタ保存領域1104と重なっており、先ほど保存したCPUレジスタの値は、実施の形態1の場合と同様に上書きされる。最後に、ステップS504にて、アプリケーションによって定義された割込みハンドラ関数をコールし、ステップS505にてアプリケーションの割込みハンドラプログラムを実行する。ステップS505からは関数リターンという形で、図8で示されるOS割込み出口処理へと移行する。

#### 【0036】

一方、OS割込み出口処理は、図8により説明した従来技術と同じ処理となるが、ステップS803で復帰するCPUレジスタの値は、r0レジスタ保存領域1107を除いて、既に上書きされているため、OS割込み入口処理時に保存しておいた値ではないことに注意を要する。但し、r0レジスタの値は、r0レジスタ保存領域1107に保存されて

いた正しい値が復帰される。もともと発生した割込みは、アイドルタスク内で r0 レジスタを使用中に受理されたものであるが、割込みから復帰した後もアイドルタスクプログラムは r0 レジスタの内容を使って、CPU を低消費動作モードへ移行させることが可能である。

#### 【0037】

以上の説明においては、アイドルタスクの実行中に割り込みが発生した場合の OS 割込み入口処理を説明したが、割込み処理スタックがアイドルタスクスタックに重畳されているので、任意のタスクの実行中に割り込みが発生した場合に、OS 割込み入口処理において実施の形態 1 で説明したものと同様の処理が行われる。すなわち、割込みが発生したタスクの CPU レジスタの内容をタスクスタックに保存した後、最終の SP 1202 の値を割込み処理スタックの SP 保存領域 1105 に保存する。

#### 【0038】

そのために、割込み処理スタックの重畳先、すなわちアイドルタスクスタックの CPU レジスタ保存領域の先頭アドレスが OS 割込み入口処理プログラムあるいは割込み制御部 1211 から与えられ、これと SP 差分定数 1250 の値を ALU 1204 で演算し、これをオペランドアドレス 1207 および BCU 1212 を経由して外部メモリ (RAM) 1214 へアドレスとして渡すことで、前記最終の SP の値を保存する。ここで、本実施の形態における SP 差分定数 1250 には、アドレス差分値として、r0 レジスタ 1107 の退避分を考慮した値を与える。その結果、割込み処理スタックの SP 保存領域 1105 の先頭アドレスを、アイドルタスクスタックにおける r0 レジスタ 1107 の退避領域の次のアドレスにすることができる。

#### 【0039】

以上のように、割込み処理スタックをアイドルタスクのタスクスタックに重畳することにより、アイドルタスクにおいて一部の CPU レジスタを使用している場合にも、アイドルタスクのタスクスタックの使用されない CPU レジスタ保存領域を有効活用することができる。

#### 【0040】

##### (実施の形態 3)

実施の形態 1 および実施の形態 2 では、アイドルタスクに注目して説明を行なったが、タスクが使用する CPU レジスタが明確に分かっている場合には、アイドルタスク以外の通常のタスクにおいても同じ手法を適用することができる。また、タスクが使用する CPU レジスタが静的に決められない場合でも、上書きしてよい CPU レジスタと上書きしてはならない CPU レジスタを、タスクの処理の中で変数などに情報を残すようにすることで、実施の形態 2 で説明した手法を同様に適用することができる。さらには、C 言語のような高級言語を使用する場合でも、コンパイラが CPU レジスタに関する情報を、変数もしくは CPU に設けられた SP 差分情報格納レジスタに保存するようにすることで、実施の形態 2 で説明した手法を適用することが可能である。

#### 【0041】

図 13 は、このようなマルチタスクシステムを実現する本発明のタスク制御装置を示している。本発明のタスク制御装置は、実施の形態 1 および 2 におけるタスク制御装置に比べて、SP 差分情報格納レジスタ 1300 を搭載している点が異なっている。SP 差分情報格納レジスタ 1300 の使用法は任意であるが、ステップ S502 で参照するタスク動作時の最終 SP 値を保存するアドレス値をコンパイラが予め算出しておき、SP 差分情報格納レジスタ 1300 へ保存できる。このようにすれば、任意のタスクの実行中に割込みが受理された場合に、OS 割込み入口処理プログラムあるいは割込み制御部 1211 に予め与えられた、割込み処理スタックの重畳先のタスクスタックにおける CPU レジスタ保存領域の先頭アドレスと SP 差分情報格納レジスタ 1300 とにより、ALU 1204 で演算した値を、オペランドアドレス 1207、BCU 1212 を経由して外部メモリ (RAM) 1214 へアドレスとして渡し、割込みが発生したタスクの CPU レジスタの内容をタスクスタックに保存した後の最終 SP 1202 の値をオペランドデータ 1208 と

して外部メモリ（RAM）1214へ書出すことが可能である。すなわち、割込み処理スタックのスタート番地をコンパイラなどがSP差分情報格納レジスタ1300に設定する任意の値で変更することが可能である。もちろん、SP差分情報格納レジスタ1300に設定するものは、コンパイラ以外の通常のアプリケーションなどのプログラムであっても構わないし、ハードウェアで設定することも可能である。設定の方法は、本発明の趣旨を越えない部分において、さまざまなものが考えられる。

#### 【0042】

以上の実施の形態において、PCとPSWの値をタスクスタック上に保存する実施例を説明してきたが、CPUの種類によってはタスク動作中のレジスタバンクと、割込み処理用のレジスタバンクを別々に搭載しているものもある。このようなCPUでは、割込み受理時にPCとPSWの値をタスクスタック上に保存せずに、割込み処理専用のレジスタバンク内にある保存領域にコピーする場合もある。この場合、割込み処理専用のレジスタバンク内に保存されたPCとPSWの値を、ソフトウェア的にタスクスタック上に保存するようにすることで、以上の実施の形態と全く同様の効果を奏することが可能である。

#### 【産業上の利用可能性】

#### 【0043】

本発明のマルチタスクシステムにおけるメモリ管理方式およびタスク制御装置は、リアルタイムOSを用いたシステムのタスクスタック領域と割込みスタック領域とを重複して使用することで、システム全体でのメモリ使用量を削減できるということが言える。したがって、小容量のメモリでマルチタスクシステムを稼働させることができるという利点があり、ソフトウェアのスタックメモリの管理方法、特にマルチタスクシステムを実行する場合のスタックメモリ使用量を削減するプログラム構造等として有用である。

#### 【図面の簡単な説明】

#### 【0044】

- 【図1】 本発明の実施例のスタック構造を模式的に示す図
- 【図2】 従来のマルチタスクシステムにおけるスタック構造を模式的に示す図
- 【図3】 従来のマルチタスクシステムにおけるスタック使用状況を詳細に示す図
- 【図4】 本発明のマルチタスクシステムにおけるスタック使用状況を詳細に示す図
- 【図5】 従来の割込み発生時のスタック切替え処理の内容を示すフローチャート
- 【図6】 本発明の割込み発生時のスタック切替え処理の内容を示すフローチャート
- 【図7】 本発明の低消費動作モードへ移行させる場合の割込み発生時のスタック切替え処理の内容を示すフローチャート
- 【図8】 割込み終了時のスタック切替え処理の内容を示すフローチャート
- 【図9】 空ループの場合の、アイドルタスク処理の内容を示すフローチャート
- 【図10】 低消費動作モードへ移行する場合の、アイドルタスク処理の内容を示すフローチャート
- 【図11】 低消費動作モードへ移行させる場合のアイドルタスク用スタックと割込み処理用スタックの構造を詳細に示す図
- 【図12】 実施の形態1および2におけるタスク制御装置
- 【図13】 実施の形態3におけるタスク制御装置

#### 【符号の説明】

#### 【0045】

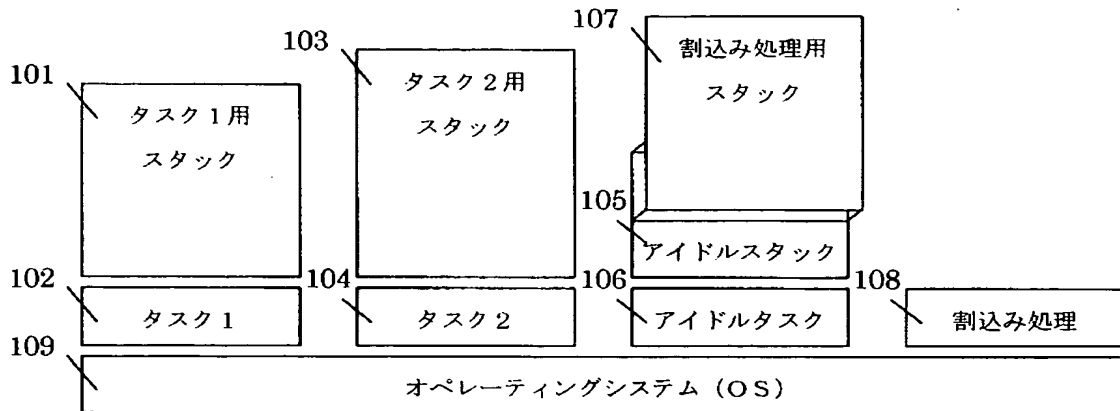
- 101 タスク1のスタック領域
- 102 タスク1のプログラム
- 103 タスク2のスタック領域
- 104 タスク2のプログラム
- 105 アイドルタスクのスタック領域
- 106 アイドルタスクのプログラム
- 107 割込み処理のスタック領域
- 108 割込み処理のプログラム

109	リアルタイムOS
205	アイドルタスクのスタック領域 (非破壊領域)
207	割込み処理のスタック領域
301	アイドルタスクが使用しているスタック領域
302	アイドルタスク動作中に割込みが発生した場合の戻りPC保存領域
303	アイドルタスク動作中に割込みが発生した場合のPSW保存領域
304	アイドルタスク動作中に割込みが発生した場合のCPUレジスタ保存領域
305	割込み発生時にそれまで動作していたタスクのSPレジスタ値を保存する領域
306	割込み処理で使用するスタック領域
311	タスク1が使用しているスタック領域
312	タスク1動作中に割込みが発生した場合の戻りPC保存領域
313	タスク1動作中に割込みが発生した場合のPSW保存領域
314	タスク1動作中に割込みが発生した場合のCPUレジスタ保存領域
321	タスク2が使用しているスタック領域
322	タスク2動作中に割込みが発生した場合の戻りPC保存領域
323	タスク2動作中に割込みが発生した場合のPSW保存領域
324	タスク2動作中に割込みが発生した場合のCPUレジスタ保存領域
401	アイドルタスクが使用しているスタック領域
402	アイドルタスク動作中に割込みが発生した場合の戻りPC保存領域
403	アイドルタスク動作中に割込みが発生した場合のPSW保存領域
404	アイドルタスク動作中に割込みが発生した場合のCPUレジスタ保存領域
405	割込み発生時にそれまで動作していたタスクのSPレジスタ値を保存する領域
406	割込み処理で使用するスタック領域
501	CPUレジスタをタスクスタックへ保存するステップ
502	タスク動作時のSPを割込みスタックのボトム位置へ退避するステップ
503	従来のSPの値を割込み処理用スタックへ切替えるステップ
504	アプリケーションの割込みハンドラを呼出すステップ
505	アプリケーションの割込みハンドラ処理を実行するステップ
603	実施の形態1におけるSPの値を割込み処理用スタックへ切替えるステップ
701	実施の形態2におけるCPUレジスタをタスクスタックへ保存するステップ
703	実施の形態2におけるSPの値を割込み処理用スタックへ切替えるステップ
801	SPの値をタスクスタックへ切替えるステップ
802	タスクスイッチが必要であれば遅延ディスパッチを行なうステップ
803	スタックに退避されているCPUレジスタ値を復帰するステップ
804	割込み処理から復帰する命令を実行するステップ
901	自分自身への空ループを行なう処理ステップ
1001	マイコンを低消費動作モードへ移行させるステップ
1101	アイドルタスクが使用しているスタック領域
1102	アイドルタスク動作中に割込みが発生した場合の戻りPC保存領域
1103	アイドルタスク動作中に割込みが発生した場合のPSW保存領域
1104	アイドルタスク動作中に割込みが発生した場合のCPUレジスタ保存領域
1105	割込み発生時にそれまで動作していたタスクのSPレジスタ値を保存する領域
1106	割込み処理で使用するスタック領域
1107	r0レジスタ保存領域
1200	実施の形態1および2におけるタスク制御装置
1201	汎用CPUレジスタ
1202	スタックポインタ (SP)

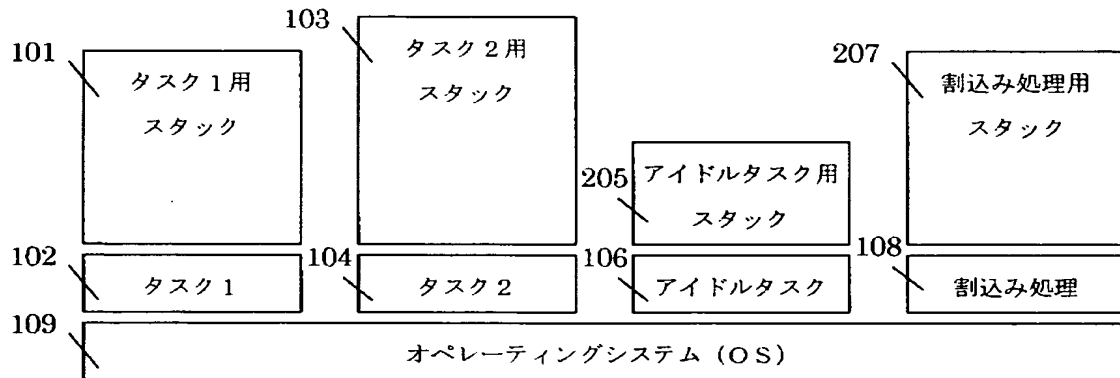
1 2 0 3	プロセッサステータスワード (P S W)
1 2 0 4	演算ユニット (A L U)
1 2 0 5	プログラムカウンタ (P C)
1 2 0 6	命令実行制御部 (命令デコーダ)
1 2 0 7	オペランドアドレス
1 2 0 8	オペランドデータ
1 2 0 9	命令アドレス
1 2 1 0	命令データ
1 2 1 1	割込み制御部
1 2 1 2	バスコントローラユニット (B C U)
1 2 1 3	読み込み専用外部メモリ (R O M)
1 2 1 4	読書き可能外部メモリ (R A M)
1 2 1 5	周辺機能
1 2 5 0	S P 差分定数
1 3 0 0	S P 差分情報格納レジスタ

【書類名】 図面

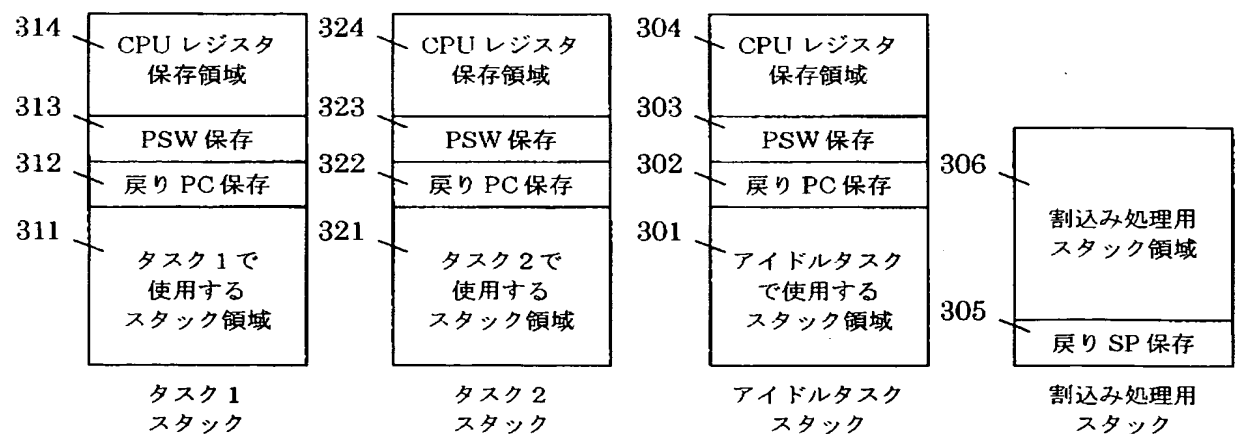
【図 1】



【図 2】

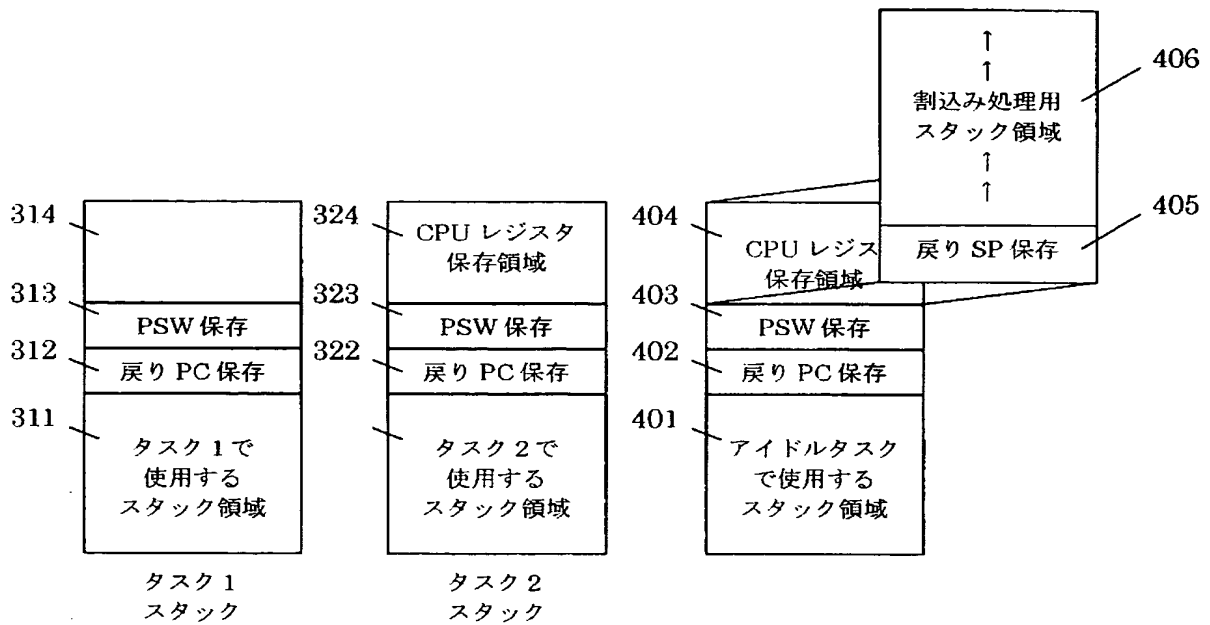


【図 3】

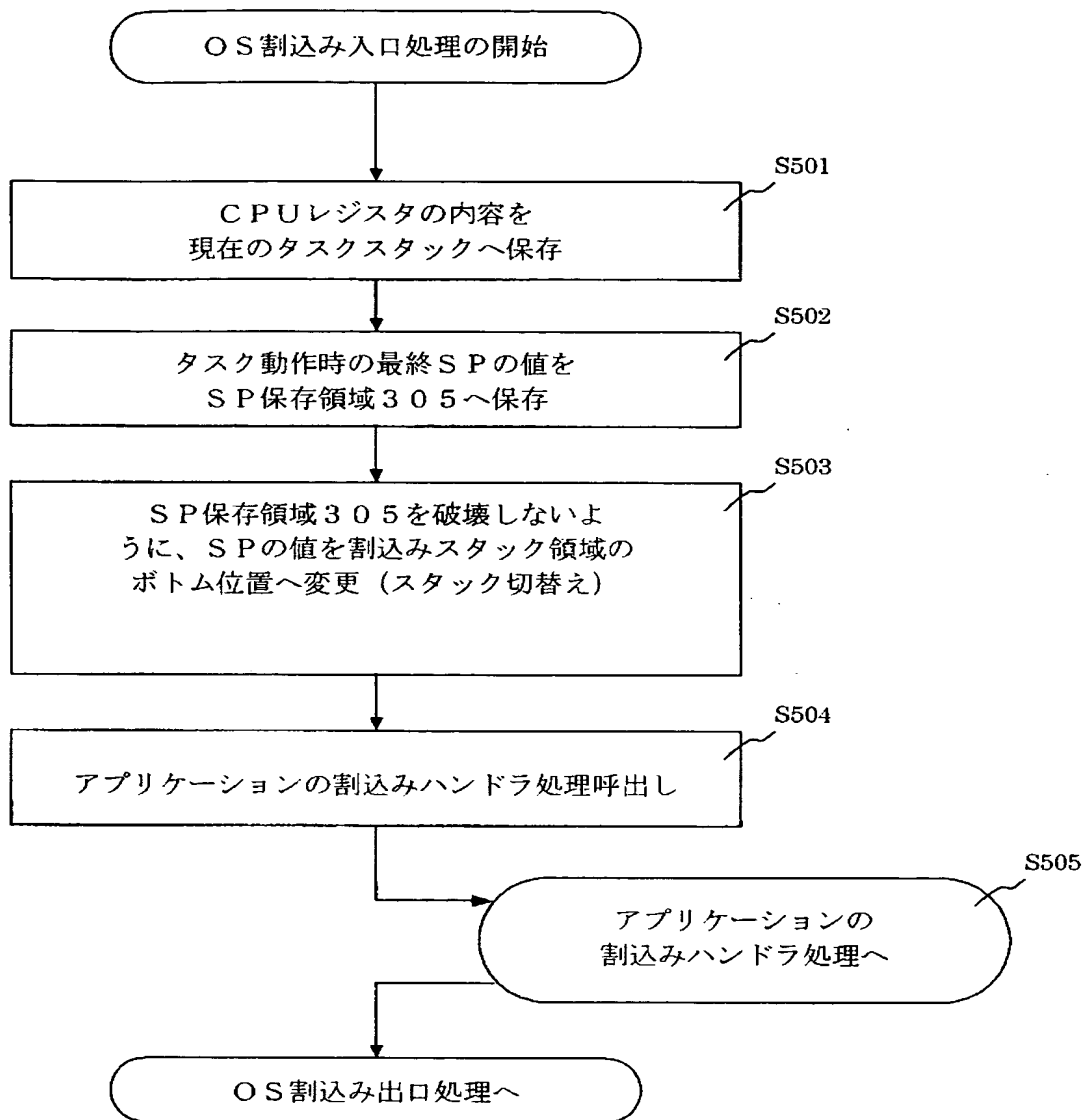




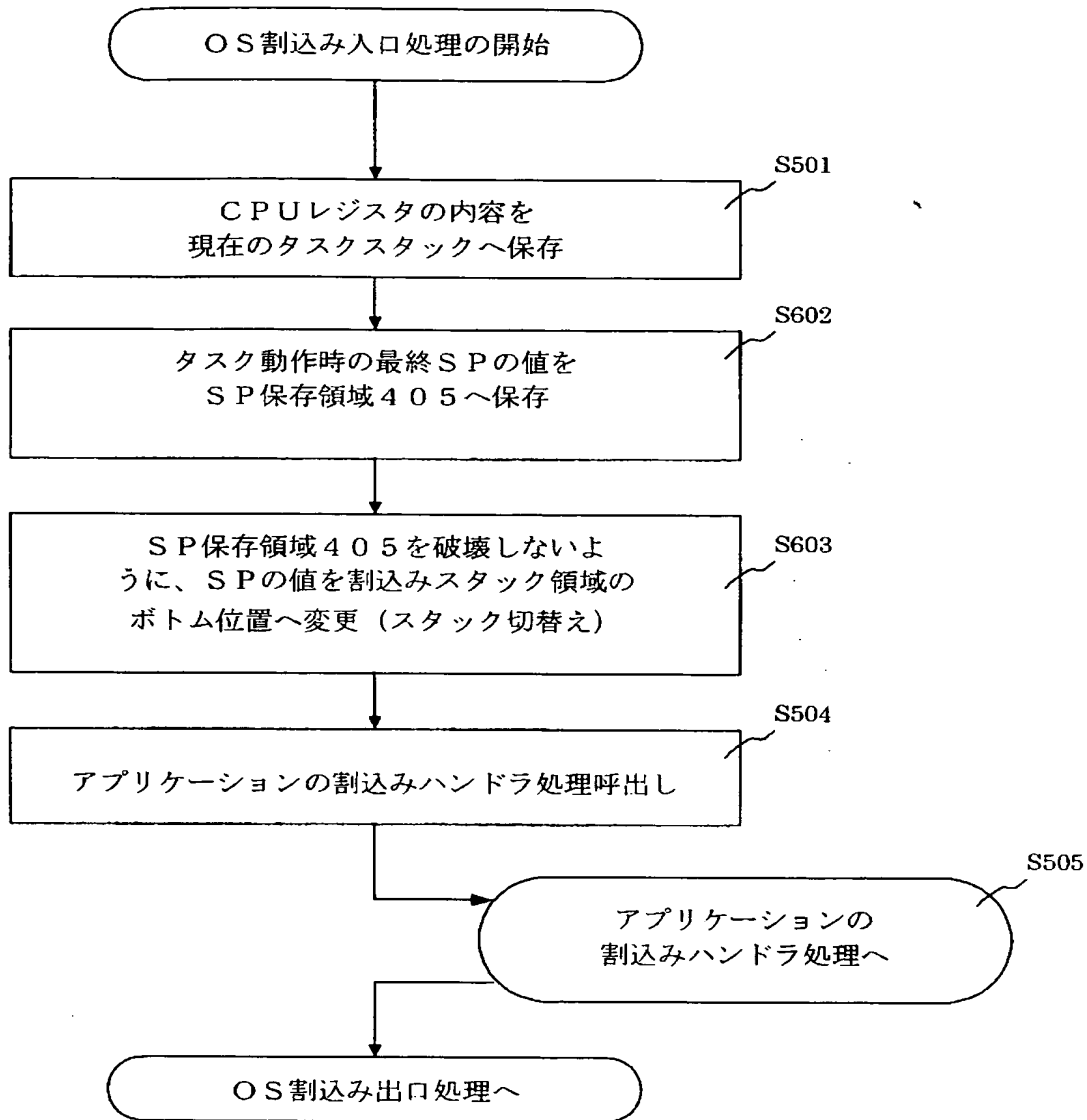
【図 4】



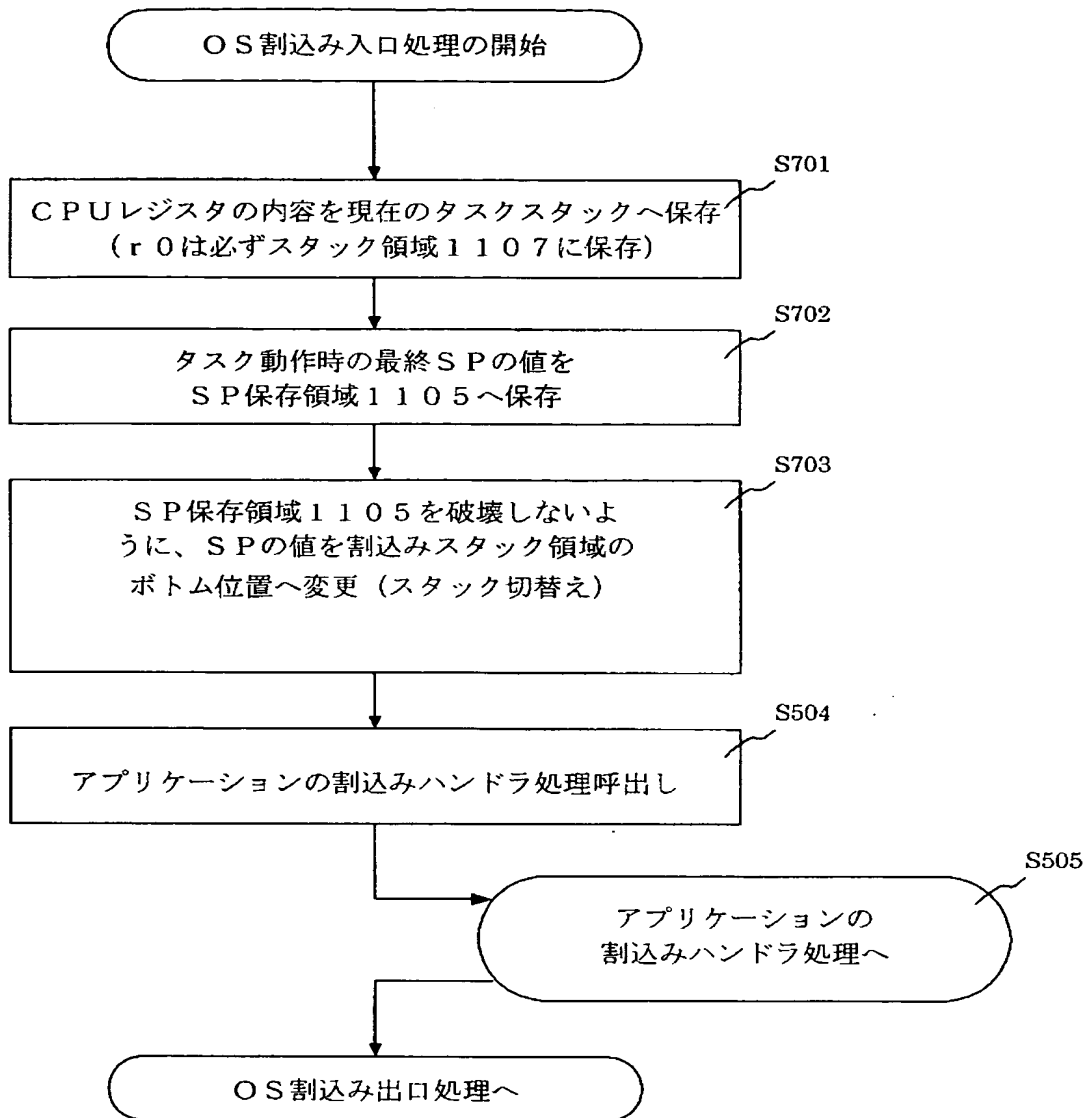
【図 5】



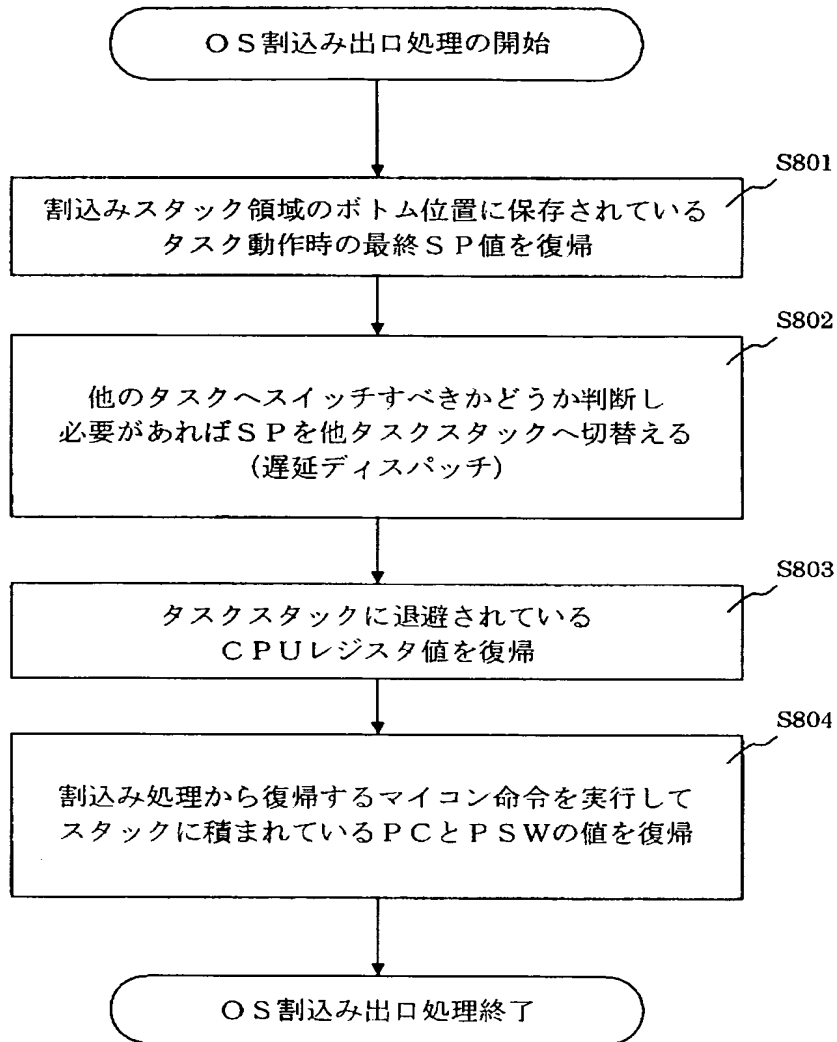
【図 6】



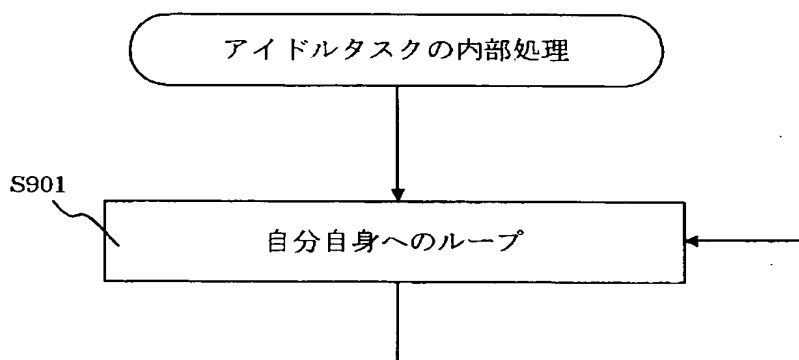
【図 7】



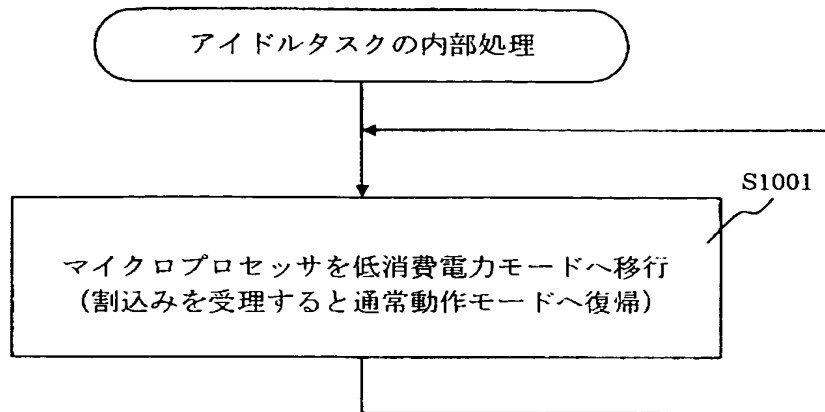
【図 8】



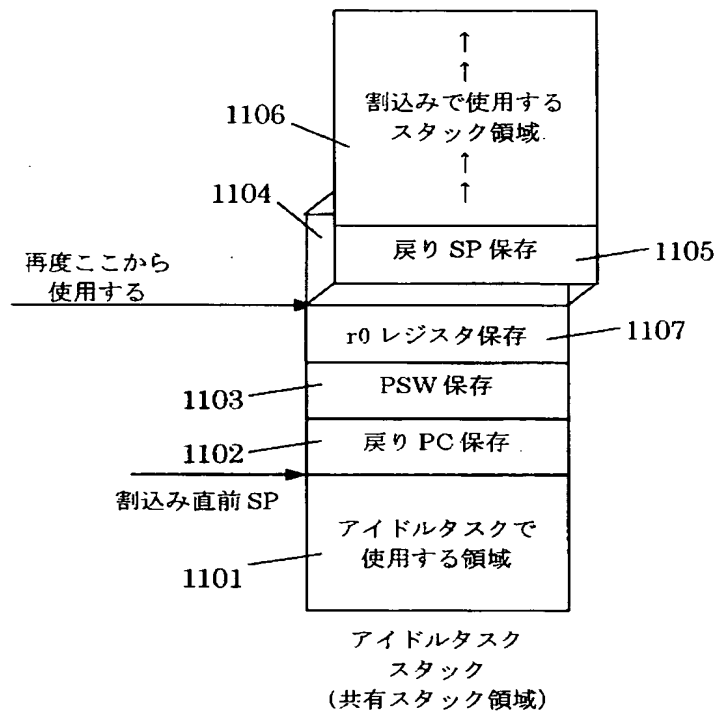
【図 9】



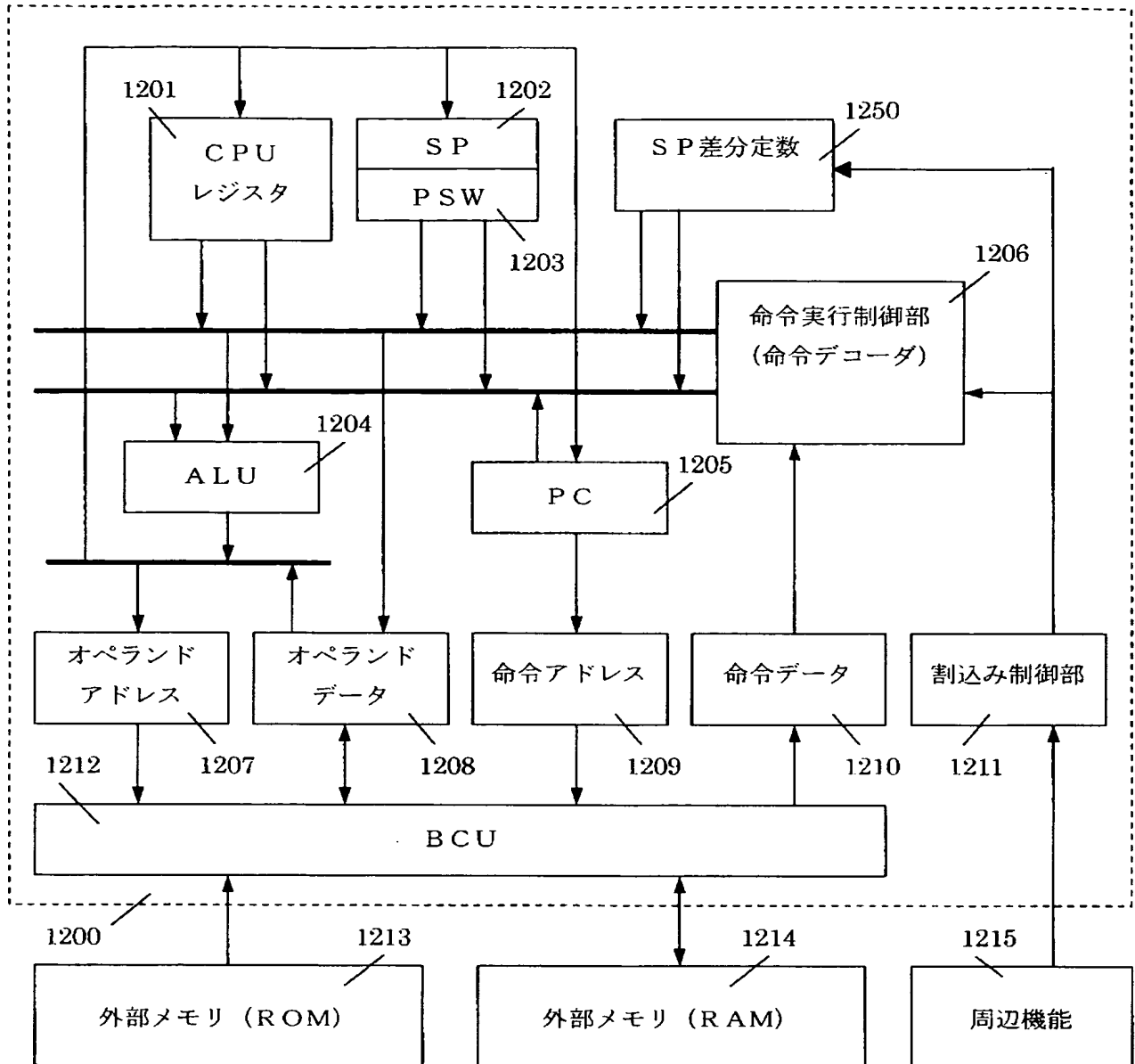
【図 10】



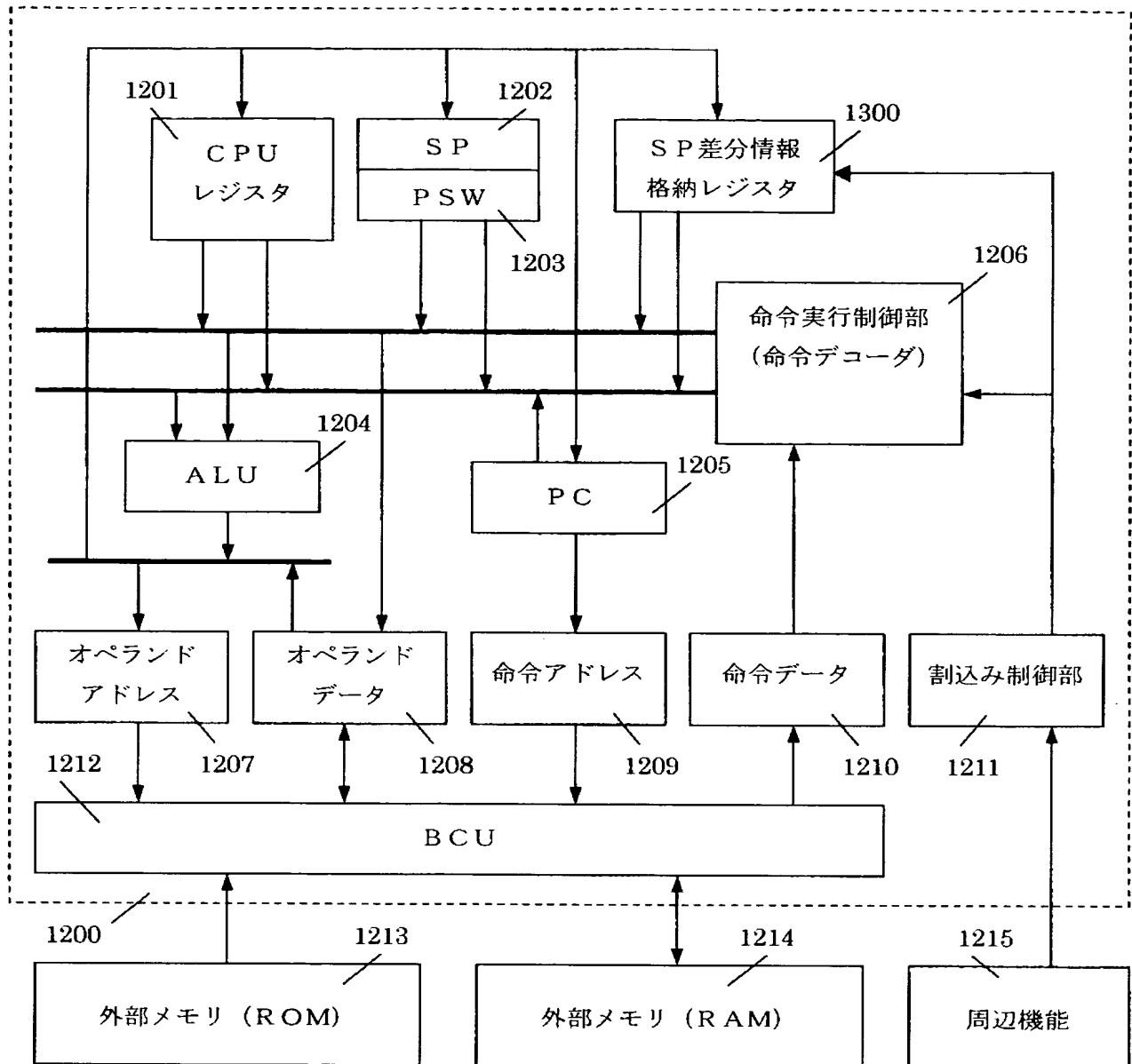
【図 11】



【図 12】



【図 13】





**【書類名】 要約書**

**【課題】** マルチタスクシステムにおいて、割込み処理が使用するスタックをアイドル処理が使用するスタックと共有することで、R A M使用量を削減する。

**【解決手段】** アイドルタスク動作中に割込みが発生すると、現在のスタック領域 1 0 5 に C P Uレジスタの値を保存した後、割込み処理専用のスタック領域 1 0 7 に切替える。このとき、スタック領域 1 0 5 と、割込み処理専用のスタック領域 1 0 7 が重なるようなスタックの構成にしておく。アイドル処理中に割込みが発生した場合は、C P Uレジスタの値を保存した領域を上書きするように割込み処理のスタックを使用する。

**【選択図】** 図 1

特願 2 0 0 3 - 2 8 2 8 8 7

出 願 人 履 歴 情 報

識別番号

[ 0 0 0 0 0 5 8 2 1 ]

1. 変更年月日

1 9 9 0 年 8 月 2 8 日

[変更理由]

新規登録

住 所

大阪府門真市大字門真 1 0 0 6 番地

氏 名

松下電器産業株式会社